LAUTERBACH
DEVELOPMENT TOOLS

# Introduction to TRACE32

## Introduction

Firstly, please accept my apologies for a somewhat lengthy read. I had not intended it to be this large but I did want this to be as comprehensive as possible in order to provide you with as much information as you need to make an informed decision about which combination of Lauterbach products best fits your needs, budget, and future plans.

At first glance, the tool offerings from Lauterbach may appear to be overwhelming in their complexity and amount of choice. Not surprising after building debug tools for 40 years! This document aims to clearly and concisely introduce the reader to the TRACE32 debug system. The TRACE32 toolset supports a large number of microprocessors, real-time operating systems, integrations, and eco-systems and this document will try to keep things as generic as possible; the details can come later when you know which questions to ask.

From the birth of the company, in 1978, the over-riding philosophy was to make modular tools that could be upgraded or expanded as required. Of course, in the real world, a step change occurs every 8-10 years as things become so much more complex or faster that a new set of tools needs to be introduced but, as far as possible, the goal of retaining as much of the initial investment still pervades everything the company does.

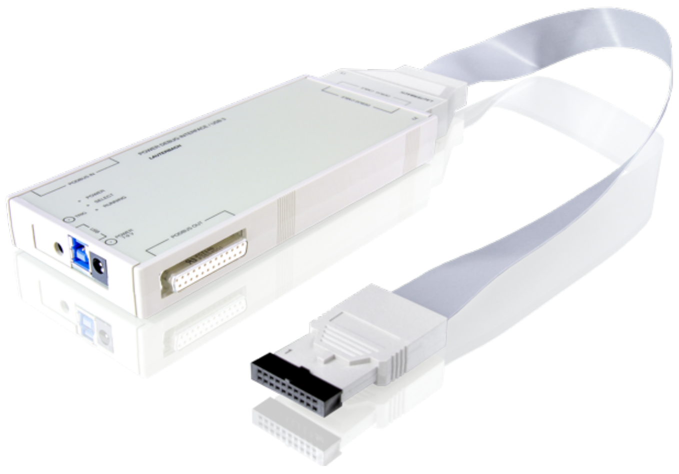
Lauterbach has always focused their attention on the debug aspect of embedded development and not tried to diversify into anything outside of their core competencies. This means that we make the best debugger we can and have to be agnostic about which compilers, IDEs, ecosystems, and RTOSes we support. As long as these follow a well-defined industry standard they are almost certain to work with your Lauterbach TRACE32 debugger.

In this document I will use the term JTAG as it is quite generic and reasonably well understood. Some devices use alternative debug connection methods: BDM, DAP, EJTAG, LDP, etc. So, where you see ‘JTAG’, please replace this with ‘debug connection that is appropriate for my target’.

I have deliberately stayed away from providing any form of numerical pricing, opting for comparative pricing between equivalent tools where there is an overlap. Pricing is subject to fluctuations in currency exchanges, supply of components, and a million other things. Having said that, prices (as far as the UK is concerned) have been very stable over the last decade or so but we live in interesting times and any numbers that are used today will be obsolete tomorrow.

## Basic JTAG Debug

The basic debug capabilities include the ability to connect to a target processor to control it (start, stop, step), and the ability to read and write memory and registers on the target device. If memory and registers can be accessed then it is possible to be able to program any on-chip peripherals and connected FLASH devices. A basic debug connection allows the user to set breakpoints which halt the program flow at precise locations and to examine the status of the target at these points. With a good, fast JTAG connection and a little bit of thought it is possible to manage some kinds of statistical profiling and runtime measurements.



A basic debug system, generally, comprises two parts: a host connection and a target specific cable/dongle. The two pieces connect together as shown in the image to the right. The system comes with a Power Supply, USB3 cable, TRACE32 software on a DVD, and an adapter for the Trigger pin.

There are two versions of the unit which provides the host connection and the current generation are named ‘PowerDebug’ and ‘PowerDebug Pro’. Functionally they are almost equal. The differences are summarised in the table below.

| | Power Debug | Power Debug Pro |
|---|---|---|
| **Part Number** | LA-3500 | LA-3505 |
| **Image** | | |
| **Connection to host** | USB3 | USB3 & Gigabit Ethernet |
| **Expansion Port** | Podbus Out only (Allows connection to Logic Analysers and synch with other Power Debug * Units) | Podbus Out **AND** PodBus express (Adds connection to Power Trace Modules) |
| **Can be expanded to add Trace Support** | No | Yes |
| **Relative Price** | ~ 60% | 100% |

## JTAG Cables

A debugger needs a JTAG cable. This provides the connection from the Interface Unit to the target development board. It also contains the license to debug a specific family of devices. A cable will look like the image here. The larger end connects to the 'Debug Out' connector on the Interface Unit and the smaller end will vary so that it fits with the target being debugged. Some cables, for example those designed for Automotive use, may come with a number of small adapter cables to convert the standard header to one of the many industry defined JTAG headers. The cost of the JTAG cable varies in proportion to the support effort required for the target device. Moving between devices is as simple as swapping a cable; the Interface Unit remains the same. The small end of the cable contains active components which allow the JTAG interface clock speed to be increased beyond where a traditional 'wiggler' style cable can manage.

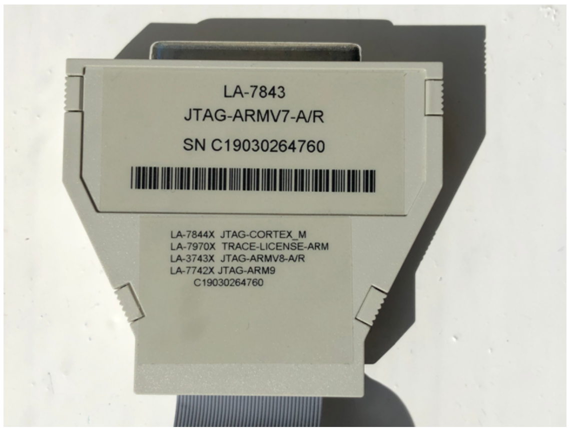
Each cable comes supplied with a base license programmed into it. Where target devices have electrically compatible JTAG interfaces, additional licenses can be programmed into the cable. A cable must be under maintenance (more about that later) before a new license can be added. Not all combinations are possible, so please contact your Lauterbach representative for assistance with this.

Two types of license can be <u>added</u> to a base cable.

Where the new license is the same family as the base license, an extension license can be added. This is the same part number as the base license but with an -X suffix. For example: Cortex-M (LA-7844) becomes LA-7844X when added to a cable with an existing **ARM** license as the base.

Where the new license is not the same family, an Additional license must be added. This is the same part number as the base license but with an -A suffix. For example: Cortex-M (LA-7844) becomes LA-7844A when added to a **non-ARM** cable, say RISC-V or ARC.

It is more cost effective to add a license to an existing cable (where possible) than to purchase a new cable. It also allows you to debug different cores. For example, a cable with a base license of ARMv8 and an additional ARC license allows you to debug any combination or ARMv8 and ARC cores which share a JTAG scan chain or reside within a single SoC. With this cable you could also debug an ARMv8 based system in the morning and then switch to an ARC based system to debug that in the afternoon.



If you order a cable with multiple licenses, they will all be listed on a sticker on the reverse of the cable. An example can be seen in the image here.

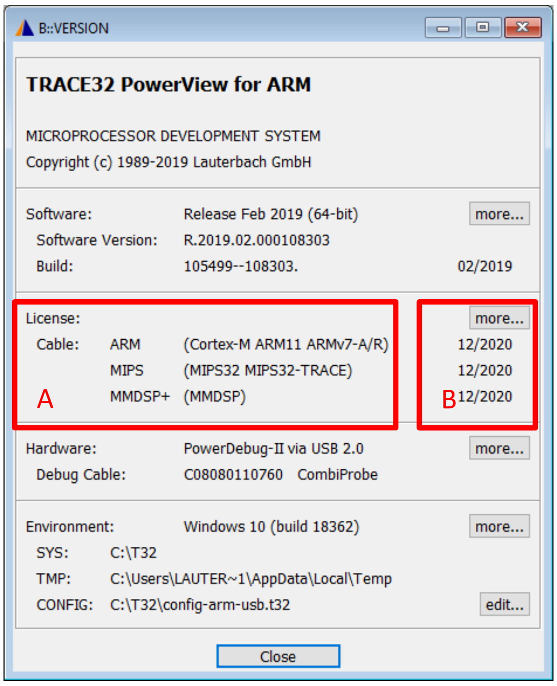
When ordering a new license for a cable, the script to program the new license to the cable is delivered by e-mail and a new sticker will follow at a later date, by post. Users in large corporations should not rely solely on the sticker as these often do not match the real capabilities of the cable; stickers can (and do) go astray!

Adding an additional (or extension) license to your cable will also unlock the multi-core features of TRACE32. A cable with only a single license (in most cases) requires an additional multi-core license to be able to debug multiple cores of the same type.

A better way of gathering data about the licenses in a cable is to use the TRACE32 software.

1) Connect the cable to an Interface Unit.
2) Connect it to a host PC.
3) Start the TRACE32 software.
4) Select **`Help->About TRACE32`**

An example can be seen in the image here.



B::VERSION

**TRACE32 PowerView for ARM**

MICROPROCESSOR DEVELOPMENT SYSTEM
Copyright (c) 1989-2019 Lauterbach GmbH

```
Software:          Release Feb 2019 (64-bit)
  Software Version: R.2019.02.000108303
  Build:            105499--108303.                    02/2019

License:
  Cable:   ARM     (Cortex-M ARM11 ARMv7-A/R)          12/2020
           MIPS    (MIPS32 MIPS32-TRACE)               12/2020
      A    MMDSP+  (MMDSP)                           B 12/2020

Hardware:          PowerDebug-II via USB 2.0
  Debug Cable:     C08080110760   CombiProbe

Environment:       Windows 10 (build 18362)
  SYS:     C:\T32
  TMP:     C:\Users\LAUTER~1\AppData\Local\Temp
  CONFIG:  C:\T32\config-arm-usb.t32
```

The list of licenses in the cable has been highlighted in region 'A'.

The expiry of the maintenance license for each of those licenses has been highlighted in region 'B'. Maintenance licenses are discussed in a later section.

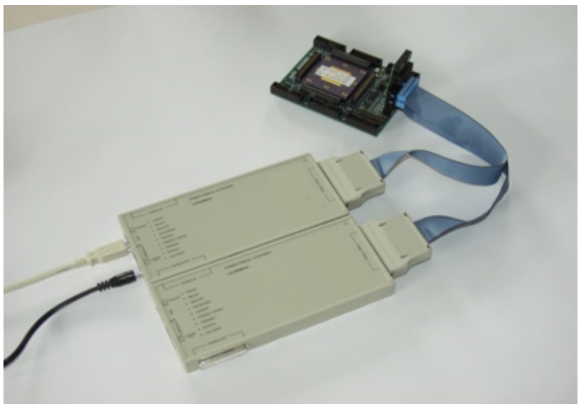
## Multi-Core Licenses

Some base licenses are multi-core enabled, for example ARMv8. Most devices that work with this type of license are only available in multi-core forms. Other licenses, for example ARMv7, are only licensed for a single core. If you wanted to debug a device with two Cortex-A9 cores in it, you would need the base license (LA-7843) plus a multi-core license (LA-7960X) in the cable.

If your development also included another family of devices, for example RISC-V (LA-2717), adding a RISC-V Additional license (LA-2717A) to the Cortex-A9 cable will also enable multi-core debugging for both the Cortex-A and the RISC-V families. We encourage you to plan ahead!

## Multi-Core Setups

TRACE32 works with all kinds of multicore configurations. Where cores are disparate, tools can be connected together via PodBus (an example can be seen in the image to the right) or via UDP through the TRACE32 software for synchronisation of start, step and stop control. This is called Asymmetric Multi-Processing (AMP).



Where a single Operating System is scheduling tasks or threads across a collection of heterogeneous processors, a single instance of multi-core licensed TRACE32 is used to control all cores. Cross-triggering is provided by on-chip resources. This is called Symmetric Multi-Processing (SMP).

## Maintenance

The software supplied with your TRACE32 system will run without restriction for the life of the tools. You are free to update your software for 12 months after the purchase date and this will also run unrestricted for the life of the tools. To update the software after this period requires a maintenance license.

- A maintenance license is required for the base license **and** each -A license in a cable.
- All -X licenses are covered by the maintenance for the base license.

| | Additional | eXtension |
|---|---|---|
| **Part Number** | LA-nnnnA | LA-nnnnX |
| **Relation to Base license** | No relation to base license, for example: ARMv8 and RISC-V | Same CPU family as base license, for example: ARMv8 and Cortex-M |
| **Maintenance** | Required | Included with base license |

A cable with ARMv8 (LA-3743), Cortex-M (LA-7844X) and RISC-V (LA-2717A) will require two maintenance licenses: (LA-3743 + LA-7844X) + (LA-2717A)

Maintenance licenses are delivered as an e-mail attachment to the person named on the Purchase Order. Please ensure we have the right address to send them to.

Maintenance licenses can be stored into the cable. A script is provided to re-program the cable to update the maintenance key.

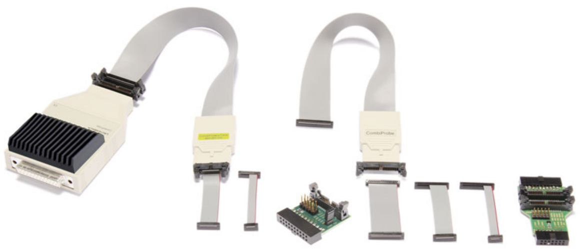
Maintenance license may also be saved into a file. The file may reside on a remote server and any user on the company network may use any cable which has a license in the central file. The file could also be stored locally but we recommend that the maintenance licenses are programmed to the cable wherever possible.

An eXtension or Additional license cannot be added to a cable unless it has valid maintenance.

Maintenance can be purchased in 12 monthly blocks, allowing you to pre-purchase 24- or 36-months' worth of maintenance for your project.

If the maintenance on a cable has lapsed, there is a discontinuity fee charged to bring the maintenance up to date before adding the new license. This is approximately 50% of the cost of an annual maintenance license.

## All-in-One Systems



Some targets are supported by a mid-range tool called a CombiProbe which provides JTAG debug and a modest trace capability with buffer sizes of  512Mbytes. An example can be seen in the image here.

It is supplied with a variety of adapters for common connectors for the CPU family.

It is not available for all targets.

It is designed to work with low to mid-range trace ports and will not cope with the fast trace ports that are supported by the Power Trace Serial or a dedicated trace pre-processor. More about these in the next section.

The supported trace ports are usually restricted width, normally 4 bits wide, for example: Cortex-M ETM/ITM, PIC32MX IFLOW trace, etc.

The CombiProbe is designed for a family of devices but the license for the specific variant required must be added. For example, the CombiProbe for ARM (LA-4597) requires an ARM family license (eg: LA-7844A for Cortex-M, or LA-7843A for Cortex-A/R) and an ARM trace license (LA-7970X).

The CombiProbe still requires an Interface Unit (LA-3500 or LA-3505) but replaces the 'traditional' JTAG cable. A system connected will look like the image here.

A CombiProbe licensed for processor trace will work with both on-chip and off-chip trace sources.

The Cortex-M family also has a very low-cost combined debug and trace unit, called the $\mu$Trace. It cannot be expanded or upgraded and only supports Cortex-M (ARMv7m and ARMv8m) based devices. It supports USB3 connection to the host and has the same debug capabilities as the Power Debug unit, and the same trace capabilities as the CombiProbe, although with only 256M trace buffer RAM. The price is a little less than the LA-3500 and doesn't require a license cable. It looks like the image here.

The unused connector (in the image) is used for an optional Analogue Probe which can be used to measure voltage and current and profile the energy use of application code. When used in conjunction with the ETM trace, the amount of power each function uses can be measured and algorithms can be adjusted to be less power hungry, if required.

## Trace Based Debug

Many modern microprocessors have a member of the family which provides a trace port. By adding extra components to the 'Pro' system, described above, it is possible to collect the trace data to give a better level of visibility into the target system. A target with a trace port, allows the processor to, non-intrusively and without altering the software, generate data about the program flow and, in some cases, data operations within the application being tested/debugged. This information is also timestamped so you can see exactly how long a section of code took to run, or how frequently it was called. Task switches can be monitored. The various ways of analysing this data are limited only by your imagination.

In addition to this, it is possible to step backwards through a sampled trace and reconstruct the context of the embedded system at any point in history, allowing you to see the root cause of a software bug or error and not just the final effect. When you run out of memory, the *malloc()* call will fail but this does not show you where the memory leak is!

Having a complete list of all instructions executed means that it is fairly trivial to collect and generate code coverage reports. TRACE32 has a Tool Qualification Support Kit (TQSK) which is free of charge to Lauterbach customers, upon registration. This kit allows developers to certify the TRACE32 tools in their environment to tool standard DO330, which means the resulting reports can be used to verify systems that must meet an industry standard safety certification.

### On-Chip Trace

For CPUs which provide trace, the trace data can generally be routed to a number of trace 'sinks'; these are points at which the trace data can be collected by a debug tool. One common option is to store the trace data in dedicated on-chip buffer RAM. On-chip resources are expensive so the trace RAM is likely to be quite small. For example, 16Kbytes on an ARMv7/v8 is typical, although some TriCore ED devices may support as much as 2Mbytes. This small buffer may be useful for debugging and catching certain kinds of problems but is seldom sufficient for long term trace analysis such as code performance metrics or code coverage.

The on-chip trace data is usually accessible via the JTAG port and requires an extra license in the JTAG cable to allow the reading and processing of this data. This is an -X license (see the previous sections for an explanation of this), for example: ARM on-chip trace license is LA-7970X, TriCore on-chip trace license is LA-3799X. The addition of an on-chip trace license does not provide multi-core support in the JTAG cable.

Without a trace license, trying to use any of the on-chip trace features will cause this part of the TRACE32 system to enter 'demo mode'. The trace features will run un-restricted for 5 minutes and then timeout, requiring a re-start of your TRACE32 system to use them again. The rest of the debug system will remain unaffected by the on-chip trace sub-system timeout. On-Chip trace licenses do not require maintenance; they are an -X license and covered by maintenance for the base license in a cable.

## Off-Chip Trace

The trace data can be sent off-chip via a dedicated trace port. These come in a variety of form factors and support many different trace protocols but all of them require a Power Debug Pro (LA-3505) as the starting point. To this a trace buffer module is added and then a trace pre-processor. Like the interface unit, the trace buffer module is designed to be re-useable and the trace pre-processor changes as you move between target processors.

The current trace buffer module is called the Power Trace 2 and comes in three varieties: 1Gbyte (LA-7692), 2Gbyte (LA-7693) and 4Gbyte (LA-7694) of buffer memory. An image can be seen in the comparison table, below.

| | Power Trace 2 | Power Trace Serial |
|---|---|---|
| **Trace technology supported** | Parallel & Serial | Serial Only |
| **Image** | | |
| **Relative Price** | 89% | 100% |
| **Target Connection** | Dedicated Pre-processor | Target Adaptation package |
| **Target Connection image** | | |
| **Change to different architecture** | Via new pre-processor | Add new license and new target adaptation package |
| **Relative cost of change** | 100% | ~45% |

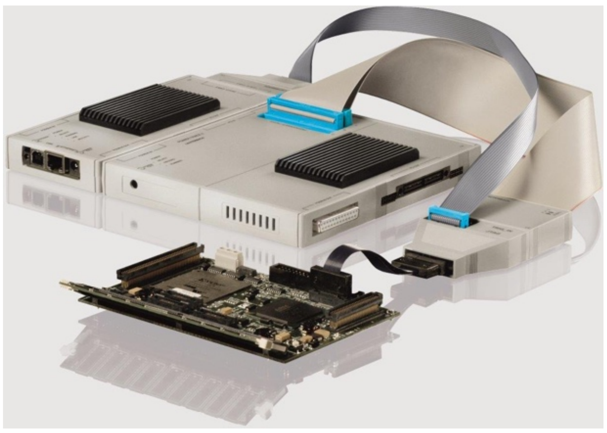
It has two PodBus Express connectors (in and out), allowing multiple Power Trace 2 units to be chained, and the trace pre-processor connects to the blue connectors on the top. The interface to the host computer is provided by the LA-3505 (Power Debug Pro). The relative costs are roughly:

- LA-7692 (68%)
- LA-7693 (76%)
- LA-7694 (100%)

The trace pre-processor can be seen in the comparison table above and is specific to a family of target processors. The idea being that if you move from debug and trace of one target to a different target, the interface unit and trace buffer are re-useable and just the pre-processor and interface cable need to change.

The pre-processor will require an adapter cable (also seen in the image) to allow it to connect to the trace port on the target development board. There is more than one standard for a trace port connector! The target specific documentation contains more information about board layout for both JTAG and trace ports.

A complete, connected system might look like the image here.



Some trace ports also carry the JTAG signals. This can save space and cost on a development board as only one header needs to be fitted. Where this is the case, the pre-processor will have a connector on it to allow you to plug-in the JTAG cable so that it can reach the JTAG pins on the trace header. This can be seen in the image to the right.

Recent developments in trace technology are moving away from parallel trace, where a trace port comprises multiple data lines and some control signals, and towards serial trace, where the trace is transmitted over a pair of pins (clock and data). This offers several advantages: fewer pins are required on the CPU package, signal skew becomes much less of an issue, trace clock speeds can become much faster, most devices already have a suitable transport interface on them (SATA, PCIe, etc). Multiple lanes can also be used, so the trace port can scale as more cores are added to the device.

The Power Trace 2 unit can be equipped with either a parallel or serial trace pre-processor. The cost of the serial pre-processor is slightly higher than that of the parallel pre-processor for the same family of devices. The other option is to use a Power Trace Serial with an appropriate target adaptation pack and license. The differences are summarized in the table on the previous page. To summarise: The Power Trace Serial differs from the PowerTrace II + Pre-processor in a number of key ways. These are summarised below:

- PT-Serial can support much higher data throughputs than PT2
- PT-Serial is a more cost-effective solution
- Since you cannot swap a pre-processor, PT-Serial loses some flexibility when compared to PT2.

## Simulators, Integrations and RTOS Awareness

Lauterbach only make debug tools and in order for them to be as generally useful as possible TRACE32 needs to be able to work with many other tools. The aim of this section is to introduce some of the capabilities; the details can follow, if required.

### Simulators

TRACE32 can be configured as an instruction set simulator. To do this requires a license. This can either be the license in a connected set of TRACE32 tools, or a software license based upon the Reprise License Manager. More information about this can be found at:

https://www.lauterbach.com/rlmhostid.html

This mode also allows TRACE32 to be used as a front end for various virtual targets or a GDB server. Again, the most current information and list of supported virtual targets can be found on the Lauterbach website:

https://www.lauterbach.com/felist.html

### Integrations

TRACE32 can integrate with a large number of other tools and development environments, for example Matlab or Eclipse. The TRACE32 software component has several open APIs (eg: C, C++, Python, VB, Java) which allows for 'glue logic' to be created to bridge to different tools. A complete, up-to-date list can always be found here:

https://www.lauterbach.com/int.html

### RTOS Awarenss

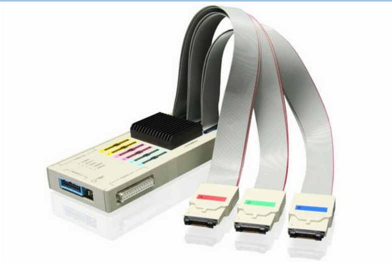
TRACE32 supports numerous Operating Systems awareness plugins. An RETOS awareness provides operating system specific features for TRACE32: the ability to display operating system objects (tasks, threads, semaphores, mailboxes, etc.), the ability to set task aware breakpoints and support for any other features that a chosen OS may provide. These are provided free of charge and are included on the software DVD that ships with each system. Updated versions can also be downloaded from the Lauterbach website. Most awareness plugins require only a debug build of the target system and rely on operating system symbols in order to function. The latest version of the list of supported RTOS types can be found here:

https://www.lauterbach.com/xlist.html

If a particular RTOS is not listed, please contact your local Lauterbach representative for more help.

## Logic Analysers

To round out the product line, Lauterbach has a pair of Logic Analysers: The Power Probe and the Power Integrator. They are designed to connect to the PodBus on the Interface Units and will also work with a Power Trace 2 or Power Trace Serial to allow synchronous recording of program flow trace and real-world signals. They can also cross trigger with the 'regular' debug system allowing for complex 'mixed' breakpoints to be set. The main differences are summarised in the table below.

| | Power Probe | Power Integrator |
|---|---|---|
| **Part Number(s)** | LA-7930 (128K frames)<br>LA-7931 (256K frames) | LA-7940 (512K frames) |
| **Image** | |  |
| **No. Channels** | 64 channels @ 100MHz<br>32 channels @ 200MHz<br>16 channels @ 400MHz | 204 channels @ 400MHz |
| **Connection** | Flying leads | Multiple target adaptation and support packages |
| **Signal Generator** | Yes<br>9bits @ 50MHz | No |
| **Probe support** | Digital and Analogue | Digital and Analogue |
| **Relative cost of change** | ~59% | 100% |

The Power Integrator supports multiple different adaptors, for example: PCIe, SODIMM, Mictor, Samtec, single ended, differential, USB2, DigiRF, etc. It can also be programmed to sample 'trace' from different interfaces, for example: RAM bus trace, USB2 trace, etc.

Both systems are fully programmable so can be adapted to suit your requirements. If you would like to know more about these options, please contact your local Lauterbach representative.

## Conclusion

I hope reading this has been a valuable use of your time. Much more information about all of this is available on the Lauterbach website (www.lauterbach.com) and now you have some search terms to use to narrow down what you want to find. Also, on the website is the complete set of documentation which can be downloaded and sample setup scripts and configurations for many common development boards.